Machine Learning Engineer Nanodegree

Qoura Question Pairs with Same Intent

Sumit Chatterjee
April 28th, 2019

Proposal

Domain Background

It is a site where people share there views with each other regarding a particular question and give a correct answer regarding a question. It's a multi-faceted kind of site, On one hand there is the academic side where people generally answers questions which are complex and also, academic where answers which, while they may have multiple correct answers, are not objective. Apart from this we also have the Meme side wich is self explanatory. People post memes for fun. Questions about other stuffs which doesn't include education, academia, religion etc. This could mean a wide range of question topics like celebrity gossips, sport related, conspiracy theories, health beneifts of certain foods, excerices etc. Quora also supports new writers.

The main reason I am choosing this particular dataset and problem and domain is because thhis problem mostly deals with words, values and strings and therefore NLP could be a great way deal this problem, and since we haven't covered much on NLP part with this nanodegree I'd like to explore this part.

Problem Statement

Accroding to Qoura over 100 million people visits Quora every month, so it's very evident that many people ask similar questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. This means

Currently, Quora is using a Random Forest model to identify duplicate questions. With this NLP problem by applying advanced techniques to classify whether question pairs are duplicates or not will make it easier to find accurate answers to questions resulting in an improved experience for Quora writers and readers as, effectively detecting duplicate questions not only saves time for seekers to find the best answer to their questions, but also reduces the effort of writers in terms of answering multiple versions of the same question. To deal with this duplicate detection problem, I am planning to apply simple feature engineering model.

Datasets and Inputs

Here is an descriptive details for the data for this capstone project proposal:

We would like to predict which of the provided pairs of questions contain two questions with the same meaning.

We have 4 data fields such as:
id - the id of a training set question pair
qid1, qid2 - unique ids of each question (only available in train.csv)
question1, question2 - the full text of each question
is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

The training data set contains 404290 question pairs, whereas the test data set has 2345790 question pairs. Each data entry consists of the ID of the question pair, the unique IDs and full text of each question, as well as the target variable mentioning whether the two questions are duplicates or not. The question pairs consists with wide range of topics which includes technology, entertainment, politics, culture, reigion, philosophy etc. Few questions also include special characters such as mathematics symbols and foreign language characters. Also, to note that the duplicate labels in the training set are provided by human experts and are inherently subjective, since the true meaning of sentences can never be known with certainty. As a result, the labels on the training data set represent a reasonable consensus and are considered ground truth, but are not 100% accurate, i.e. the dataset is noisy.

Solution Statement

In this since it is a NLP based problem I'd start by performing data cleansing, which involves removing the stop words, replacing the abbreviations with the corresponding full words, and converting all the text into lower-case. The feature engineering with respect to the simple word-level and sentence-level features will be done. It'll be where it is evident that the two features with respect to the difference in question lengths as well as the difference in the total TF-IDF scores of questions hardly separate the duplicated question pairs from the non-duplicated ones. Post that I'd apply a pipeline or a set of algortihms like logistic regression, CART algorithms, XGBoost etc. The data will be split into 60-20-20 set i.e. 60% training, 20% validation and test set each.

Benchmark Model

With the experience that I have gathered and based on the problem I'd like to use a tree type model to choose my benchmark model i.e. extreme Gradient Boosting, and will use hyperparameter tuning to beat the accuracy of the model.

Evaluation Metrics

The evalutaion metrics that I'd be using are the three pillars of any DS project namely accuracy, sensitivity & specivity, and it uses True Positive Rate(TPR), True Negative Rate(TNR), False Positive Rate(FPR), False Negative Rate(FNR).

Positive(P):The number of real positive cases in the data;
Negative(N):The number of real negative cases in the data;

therefore, our TPR looks like this: TP/P, where TP is true positive i.e. predicted positive.
TNR looks like this: TN/N, where TN is true negative i.e. predicted negative.
FPR looks like this: FP/N, where FP is false positive, i.e. false alarm.
FNR looks like this: FN/P, where FN is flase negative, i.e. miss
.

I'd also calculate the **Accuracy** which can be done by: **(TP+TN)/(TPR+TNR+FPR+FNR)**

Precision is also one more evaluation metric that is defined as number of true positives(TPR) by the number of true positive and false positive: **TPR/TPR+FPR**

Recall is also an evaluation metric that is defined as number of true positives(TPR) by the number of true positive and false negatives: **TPR/TPR+FNR**

Project Design

Project design is as follows:

1. Exploring the data, taking basic statistics out of the data.

```
 - Importing libraries and data.
 - Checking the data and picking the nitty gritty of the data, makin
g basic visualization to understand the data better.
 - Taking basic statistic out of the data.
```

2. Data cleansing and pre processing.

```
 - Preprocess feature columns.
 - Identify feature and target columns.
 - Data Cleaning to make (e.g. removing stop words, making all words
small etc)
 - Training, Validation & testing data split.
 - Feature scaling - Standardization/Normalizing data.
```

3. Evaluate Algorithms.

```
 - Build models (Using different model and also extreme XGBoosting).
 - Select best model (Using hyperparameter tuning and grid search).
 - Making predictions on the validation set.
 - Feature importance and feature selection.
```

4. Model Tuning to improve result.
5. Final conclusion.

Resources

1. https://www.kaggle.com/c/quora-question-pairs/data (https://www.kaggle.com/c/quora-question-pairs/data)
2. https://www.kaggle.com/c/quora-question-pairs/overview (https://www.kaggle.com/c/quora-question-pairs/overview)
3. http://xiaojizhang.com/files/quora-question-pairs.pdf (http://xiaojizhang.com/files/quora-question-pairs.pdf)
4. https://en.wikipedia.org/wiki/Confusion_matrix (https://en.wikipedia.org/wiki/Confusion_matrix)
5. https://en.wikipedia.org/wiki/Type_I_and_type_II_errors#False_positive_and_false_negative_rat (https://en.wikipedia.org/wiki/Type_I_and_type_II_errors#False_positive_and_false_negative_ra
6. https://en.wikipedia.org/wiki/Feature_scaling (https://en.wikipedia.org/wiki/Feature_scaling)